08

Exclusión logica #2

Transcripción

[00:00] Porque cuando yo estoy haciendo aquí el medico.activo = false; estoy dejando de trabajar a nivel de la entidad de JPA. Entonces lo que yo aquí debo hacer es decirle que this.activo = false; y no voy a usar ningún parámetro aquí. ¿Por qué?

[00:19] Vamos a ver cómo es la relación que tiene JPA con esto. Voy a guardar, vemos que tengo mi desactivarMedico, va a dar un error de compilación, porque aquí está con el parámetro. Este médico de aquí ya no tiene por qué existir, porque yo ya lo tengo aquí a nivel del repositorio, como yo ya lo cargué a nivel de mi repositorio, yo ya estoy trabajando con la entidad directamente aquí.

[00:42] Yo voy a desactivar este médico que tengo acá. Entonces ahora le voy a dar a guardar. JPA en su contexto Hibernate, bueno que es la implementación, en su contexto él ya lo va a actualizar y como está con @Transactional, ahora sí, una vez que sea actualizado él ya lo va a commitear en su base de datos.

[01:03] Nuevamente delete médico, enviar, 200, actualizar médico, perdón, vamos al listado, vemos que sigue apareciendo, pero en la tabla médicos ya activo, ya está con 0. Ya está marcado con 0. ¿Esto por qué? El médico ya fue desactivado, pero ahora yo obviamente no quiero mostrar solamente mis médicos activos e inactivos. El listado debe retornar solo médicos activos según mi requerimiento inicial del negocio.

[01:36] ¿Cómo modifico esto? Bueno. Vamos aquí a mi método de listarMedico y aquí va a ocurrir algo muy curioso, yo voy a duplicar esta línea y se las voy a dejar comentada aquí para que quede como referencia. Pero voy a cerrar esto. Yo voy a decirle aquí findByActivoTrue. ¿Qué significa eso?

[02:04] Le estoy diciendo medicoRepository, encuentra por parámetro activo true. Esta nomenclatura de aquí es la que usa Spring Data, Spring JPA para crear dinámicamente las queries y hacer el where en el select, select from where activo = true.

[02:25] Esa nomenclatura es propia de Spring y nos facilita la vida. ¿Ahora por qué está dando rojo? Porque obviamente no existe en nuestra interfaz que estamos extendiendo. Esto es muy customizado digamos, yo estoy customizando de un atributo de mi propia tabla que yo lo acabo de crear.

[02:43] ¿Qué hago? Voy a crear el método. Pero yo le voy a cambiar el tipo de dato a una página de médico. Y con esto estoy diciendo: retórname una página de médicos que es esta de aquí. Regreso a lo anterior y vemos que ya está compilando. Voy a guardar. Vamos a esperar a que cargue nuestro servidor. Vemos que terminó de cargar.

[03:18] Vamos a borrar nuestra terminal y vamos a ejecutar nuevamente nuestro listado. Y como vemos el id 5 ya no se muestra más porque está desactivado. El id 5 ya no está, pero si vamos a ver a la base de datos, listamos los médicos. Vemos que en la base de datos todavía permanece. ¿Entonces, qué ocurrió aquí?

[03:39] Un delete lógico. Y vamos a ver la query. Vamos a ver qué ejecutó por ejemplo, Hibernate. Hizo un select from where, where activo y con el limit por la paginación. Entonces de esta forma ya vimos cómo Hibernate y la implementación de Spring data nos permiten incluso hacer queries personalizadas, hacer un findByActivoTrue es una query personalizada.

[04:09] Dependiendo de esa nomenclatura Spring Data va a mapear esto y va a transformarlo a una query SQL normal como la conocemos, la cual está impresa aquí abajo. Esta fue al final la query que ejecutó Hibernate. Como siempre les digo, practiquen mucho, pueden incluso para jugar, para divertirse un poco, pueden intentar hacer findByNombre y ponerle digamos otro tributo aquí, un parámetro.

[04:41] Spring es muy flexible y se presta para casi todos los casos de uso. Sin más que decirles, practiquen mucho, nos vemos en el siguiente video.